INFORMATION AND CONTROL 52, 36–51 (1982)

Analogues of Semirecursive Sets and Effective Reducibilities to the Study of *NP* Complexity*

ALAN L. SELMAN

Computer Science Department, Iowa State University, Ames, Iowa 50011

1. INTRODUCTION

The analogies between polynomial time complexity and classical recursion theory are numerous, and some of these are obvious while others run deep. A language is feasibly computable if it belongs to P; a language is computable if it is recursive. The languages in NP are those that can be expressed in the form

$$\exists y R(x, y),$$

where R is a relation in P and, for some polynomial p, y ranges over words of length not exceeding $p(|x|)$. This same analogy to the classical characterization of the recursively enumerable sets extends to the polynomial hierarchy, which of course is the natural analogue of the arithmetical hierarchy. Even Wrathall's result (Wrathall, 1976) that B_ω (a well-known complete set in PSPACE, see Stockmeyer (1976) is not in PH, if the hierarchy is proper, is reminiscent of Tarski's theorem that truth is not arithmetic.

Efficient reducibilities are used to classify decidable problems in much the same way that effective reducibilities are used to classify undecidable problems. The reducibilities formulated by Cook (1971) and by Karp (1973) are just the restrictions to polynomial time of Turing and many–one reducibilities, respectively. These are denoted $\leqslant_T^P$ and $\leqslant_m^P$, where the superscript indicates the time bound. Surprisingly, the structure of $NP \leqslant_T^P$-degrees is similar to the one of r.e. $\leqslant_T$-degrees, see Ladner (1975). For example, they form a dense uppersemilattice.

Clearly, $A \leqslant_m^P B \Rightarrow A \leqslant_T^P B$. $A \leqslant_T^P B$ does not imply $A \leqslant_m^P B$ for recursive sets that are recognizable in time 2^n (Ladner, Lynch, and Selman, 1975). It

*This research was supported in part by the National Science Foundation under Grant MSC77-23493 A02. The manuscript was prepared while the author visited the Computer Science Department, Technion, Haifa, Israel, with funds provided by the United States–Israel Educational Foundation (Fulbright Award).

0019-9958/82/010036–16$02.00/0

has been an open problem whether $\leqslant_T^P$ and $\leqslant_m^P$ differ on *NP*. We would especially like to know whether there are $\leqslant_T^P$-complete sets in *NP* that are not already $\leqslant_m^P$-complete. These are the sorts of problems that this paper will analyze.

If we are to borrow the methods of recursion theory in our study of polynomial time complexity, we must be aware that the analogies we have been drawing are not quite accurate. It is well known that the analogue of Post's theorem fails; there exist recursive sets A and B such that A and $\bar{A}$ are both *NP* in B, but A is not deterministically reducible to B in polynomial time (Baker, Gill, and Solovay, 1975). Another notable exception is given by Breidbart (1978) who shows that there can exist no infinite, coinfinite maximal languages in *NP* (in the lattice under inclusion modulo finite sets). As Breidbart says, "in this respect, nondeterministic complexity classes resemble... the recursive sets more than they resemble the recursively enumerable sets."

If $\leqslant_T^P$ and $\leqslant_m^P$ are not identical on *NP*, then $P \neq NP$. If we assume $P \neq NP$, then we still have difficulty distinguishing reducibilities on *NP*. One reason is that we do not know how to construct sets in *NP* by diagonalization. A more promising approach is given by the use of *semirecursive* sets, which have been used successfully to distinguish reducibilities on the recursively enumerable sets. What is most outstanding about the method of semirecursive sets is the fact that we can uncover structure of *NP* without the use of diagonalization. Rather than restrict our attention to many-one and Turing reducibilities (see Selman (1979)), it will be more interesting to study positive vs. truth-table reducibilities. Our principal concern is with the McLaughlin and Martin construction that appears in Jockusch (1968). As we proceed we will see that some properties of this construction have exact analogues on *NP* while others fail altogether.

It is assumed that the reader is familiar with the basic concepts of elementary recursive function theory, and with the basic concepts and issues of automata-based complexity theory. The next section reviews some of the concepts that are most important for this paper, defines polynomial time-bounded positive and truth-table reducibilities, and shows the relationships between these reducibilities and the corresponding reducibilities of recursion theory. Section 3, then, presents the polynomial time analogue of semirecursive sets, which we call "p-selective sets." Section 4 contains the main results of this paper. The McLaughlin and Martin construction in Jockusch (1968) uses semi-recursive sets to show, in an elegant way, the existence of $\leqslant_{tt}$-complete r.e. sets that are not $\leqslant_{ptt}$-complete. We will see this construction and the modifications that make it work in the polynomial time setting. We will see that under reasonable hypotheses, polynomial time positive and truth-table reducibilities can be distinguished within *NP* using the notion of p-selective sets. However, we will also see that this analogy

between recursion theory and complexity theory breaks down—this construction cannot be applied to give information about complete sets in *NP*.

2. Effective and Efficient Reducibilities

Unless specified otherwise, all sets are languages over the finite alphabet $\Sigma = \{0, 1\}$. The length of a string x in Σ^* is denoted $|x|$. Natural numbers are identified with their binary representations. Whenever a computation produces objects of some type other than words (for example, graphs, finite functions, etc.) it must be the case that these objects are efficiently encoded as words over an output alphabet. We leave choice of output language and encoding unspecified.

Let f be a function defined on the set of natural numbers. For a Turing acceptor M, $L(M)$ is the set of strings accepted by M. A Turing acceptor M, possibly nondeterministic, operates within time f if every computation of M on x, for each input word x, halts within $f(|x|)$ moves. $\mathrm{DTIME}(f) = \{L(M) | M$ is a deterministic Turing acceptor which operates within time bound $f\}$ and $\mathrm{NTIME}(f) = \{L(M) | M$ is a nondeterministic Turing acceptor which operates within time $f\}$.

The specific complexity classes we are concerned with are:

(i) the class of sets accepted by deterministic Turing machines which operate in polynomial time, $P = \mathrm{U}\{\mathrm{DTIME}(p) | p$ is a polynomial$\}$, and the corresponding nondeterministic class, $NP = \bigcup\{\mathrm{NTIME}(p) | p$ is a polynomial$\}$;

(ii) the class of sets accepted by deterministic Turing machines which operate in exponential time, $\mathrm{DEXT} = \bigcup_{c=1}^{\infty} \mathrm{DTIME}(2^{cn})$, and the corresponding nondeterministic class $\mathrm{NEXT} = \bigcup_{c=1}^{\infty} \mathrm{NTIME}(2^{cn})$.

An *oracle* Turing machine is a Turing machine acceptor with a distinguished oracle tape and three special states Q, YES, and NO. When the Turing machine enters state Q the next state is YES or NO depending on whether or not the word currently written on the oracle tape belongs to the oracle set. In this way, if the machine is provided with an oracle B, it receives an answer to a Boolean test of the form "$x \in B$" in one move. $L(M, B)$ will denote the set accepted by the oracle Turing machine M with B as its oracle. An oracle Turing machine M is said to *operate in polynomial time* if there exists a polynomial p such that for every input word x and every choice of oracle B, M halts on input x with B as its oracle within $p(|x|)$ steps. Given a polynomial $p(n) = n^c$ there is a Turing machine M_p that on every input of length n will execute for exactly n^c steps. Given an oracle Turing machine M together with a polynomial "clock" M_p, it is possible to

design an oracle Turing machine M' that simulates M and uses M_p to quarantee that M is not simulated on an input x for more than $|x|^c$ steps. Oracle Turing machines and polynomial clocks are both syntactic objects. By enumerating ordered pairs of these objects we arrive at an effective enumeration of all oracle Turing machines that operate in polynomial time. *A is Turing reducible to B* ($A \leqslant_T B$) if $A = L(M, B)$ for an oracle Turing machine that halts on every input. *A is Turing reducible to B in polynomial time* ($A \leqslant_T^P B$) if in addition M operates in polynomial time.

Given a polynomial time-bounded reducibility $\leqslant_r^P$, a set A is *$\leqslant_r^P$-hard* if, for every set L in NP, $L \leqslant_r^P A$. A is *$\leqslant_r^P$-complete* if $A \in NP$ and A is $\leqslant_r^P$-hard. In the literature (Garey and Johnson, 1979) "NP-hard" means $\leqslant_T^P$-hard and "NP-complete" means $\leqslant_m^P$-complete.

Let C_B denote the characteristic function of the set B. Recall that a set A is truth-table reducible to a set B ($A \leqslant_{\text{tt}} B$) if there is a recursive function f that on input x computes a list of queries $q_1, \ldots, q_k$ and a Boolean function α so that $x \in A$ if and only if $\alpha(C_B(q_1), \ldots, C_B(q_k)) = 1$. The definition is of course due to Post (1944).

By analogy we want to say that A is *truth-table reducible to B in polynomial time* if (i) $q_1, \ldots, q_k$ and α are computable from x in polynomial time, and (ii) for any sequence of truth-values $\sigma_1, \ldots, \sigma_k$, $\alpha(\sigma_1, \ldots, \sigma_k)$ can be evaluated in polynomial time. But to be precise we do need to worry about what language is used to encode Boolean functions, since the length of α now becomes an important issue. The following definition is from Ladner, Lynch, and Selman (1975).

Let Δ be a fixed finite alphabet for encoding Boolean functions and let $c \notin \Delta \cup \{0, 1\}$.

DEFINITION 1. A *is truth-table reducible to B in polynomial time* ($A \leqslant_{tt}^P B$) if there exist functions f, g, and e so that each of the following holds.

(i) $f: \{0, 1\}^* \to (c\{0, 1\}^*)^*$. Thus, $f(x) = cq_1 \cdots cq_k$ is a list of queries.

(ii) $g: \{0, 1\} \to \Delta^*$. Thus, $g(x)$ is an encoding of a Boolean function.

(iii) $e: \Delta^* c\{0, 1\}^* \to \{0, 1\}$. For any $\sigma_1 \cdots \sigma_k$ in $\{0, 1\}^*$, $e(g(x)\, c\sigma_1 \cdots \sigma_k)$ evaluates $g(x)$ on input $\sigma_1, \ldots, \sigma_k$ to either 0 or 1.

(iv) $x \in A$ if and only if $e(g(x)\, cC_B(q_1) \cdots C_B(q_k)) = 1$, where $f(x) = cq_1 \cdots cq_k$.

THEOREM 1 (Ladner, Lynch, and Selman, 1975). *$A \leqslant_{\text{tt}}^P B$ if and only if*

(i) *there is an oracle Turing machine M that operates in polynomial time such that $A = L(M, B)$, and*

(ii) *there is a polynomial time computable function* $f: \{0, 1\}^* \to (c\{0, 1\}^*)^*$ *such that, for each input* x *to* M, M *only makes queries to* B *from the list* $f(x)$.

Proof. Assume $A \leqslant_{\text{tt}}^{P} B$. Conditions (i) and (ii) of the theorem will be satisfied by an oracle Turing machine M that simulates (ii) through (iv) of Definition 1 and that operates in polynomial time.

Assume M and f satisfy the conditions of the theorem. Define a Boolean function $\alpha_x: \{0, 1\}^k \to \{0, 1\}$ by the following procedure.

To determine $\alpha_x(\sigma_1,..., \sigma_k)$, where each $\sigma_i \in \{0, 1\}$, simulate M on input x and whenever M enters the query state Q with q_i on the query tape set the next state to YES if $\sigma_i = 1$ and to NO if $\sigma_i = 0$. Then, define $\alpha_x(\sigma_1,..., \sigma_k) = 1$ if M accepts x in this simulation and define $\alpha_x(\sigma_1,..., \sigma_k) = 0$, otherwise.

Formally, $\Delta = \{0, 1\}$, $g(x) = x$, and $e(xc\sigma_1 \ldots \sigma_k) = \alpha_x(\sigma_1,..., \sigma_k)$ is given by the procedure. It is easy to see that $A \leqslant_{\text{tt}}^{P} B$ via f, g, and e. ∎

Observe that both $\leqslant_{\text{tt}}$ and $\leqslant_{\text{tt}}^{P}$ are defined by syntactic objects. Therefore both kinds of reductions can be effectively enumerated.

By definition, A *is positive truth-table reducible to* B ($A \leqslant_{\text{ptt}} B$) if $A \leqslant_{\text{tt}} B$ and if in addition the Boolean function α that is computed on input x can be written as a disjunction of conjunctions of propositional letters. This is equivalent to saying that α can be represented by a *positive* formula of propositional logic, i.e., using only conjunctions and disjunctions as connectives. (For notational ease, let us identify Boolean functions with their representing formulas.) Furthermore, a formula α is positive if and only if it satisfies the semantic condition: if $\alpha(\sigma_1,..., \sigma_k) = 1$ and $\rho_1 \cdots \rho_k$ is a string in $\{0, 1\}^*$ such that $\sigma_i = 1$ implies $\rho_i = 1$, for $i = 1,..., k$, then $\alpha(\rho_1,..., \rho_k) = 1$. Therefore, if α is positive, then if $\alpha(C_B(q_1),..., C_B(q_k)) = 1$ for a list of queries $q_1,..., q_k$ and $B_0 \supseteq B$, then $\alpha(C_{B_0}(q_1),..., C_{B_0}(q_k)) = 1$.

Since writing a Boolean function using only conjunctions and disjunctions may change the length of its original presentation, we use the semantic condition held by positive formulas for our complexity theoretic formulations of positiveness, rather than a syntactic approach.

DEFINITION 2. An oracle Turing machine M is *positive* if, for each input word x and oracle set B, if M accepts x with oracle B and if $B_0 \supseteq B$, then M accepts x with oracle B_0. A set A is *polynomial time positive reducible to* B, $A \leqslant_{\text{pos}}^{P} B$, if A is Turing reducible to B in polynomial time by a positive oracle Turing machine.

The reader should observe that the semantic condition imposed here is the exact semantic condition given above for positive formulas. Theorem 5 below will expand further the connection between positive machines and positive formulas.

The following properties of $\leqslant^P_{\text{pos}}$ are proved in Selman (1982).

THEOREM 2. *$A \leqslant^P_{\text{pos}} B$ and $B \in NP$ implies $A \in NP$.*

Proof sketch. Let M_1 be an NP-acceptor for B and let M_2 be a positive query machine that accepts A with oracle B. An NP-acceptor M is defined for A as follows: On input x to M, M simulates M_2, except that queries w to the oracle B are replaced by simulations of M_1 on input w. M continues its simulation of M_2 in the YES state if M accepts w, and M continues its simulation of M_2 in the NO state if M_1 does not accept w.

If $x \in A$, then M_2 with oracle B accepts x. M can correctly simulate M_2 by choosing a correct (that is, accepting) computation of M_1 for each query w which does belong to B. Thus, if $x \in A$, then some computation of M will accept x.

A computation of nondeterministic machine M_1 may fail to accept an input w that belongs to $L(M_2) = B$. Therefore, it is possible that an accepting computation of M on an input x is not an accurate simulation of M_1 on x with B as the oracle. However, it is not hard to see that if M accepts x, then M_1 accepts x with some oracle B_1 such that $B_1 \subseteq B$. Since M_1 is a positive query machine, we conclude that M_1 accepts x with oracle B. Thus, $x \in A$. Therefore, $x \in A$ if and only if M accepts x. ∎

It follows from Theorem 1 that $\leqslant^P_{\text{pos}}$ is different than $\leqslant^P_{\text{tt}}$ on NP if NP is not closed under complements.

THEOREM 3. *There exist recursive sets A and B so that $A \leqslant^P_{\text{pos}} B$ and $A \not\leqslant^P_{\text{tt}} B$.*

This result is proved by the same sort of diagonalization used in Ladner, Lynch, and Selman (1975) to obtain $A \leqslant^P_T B$ and $A \not\leqslant^P_{\text{tt}} B$. It is also true that $A \leqslant^P_{\text{tt}} B$ does not imply $A \leqslant^P_{\text{pos}} B$. Since $\leqslant^P_{\text{pos}}$ is not defined by syntactic objects, a diagonalization must necessarily be over some class that properly includes all $\leqslant^P_{\text{pos}}$ reduction procedures. Long (in press, Corollary 3.19) gives the basis of an appropriate construction. Properties of semirecursive sets will yield this result without use of diagonalization.

The characterization of $\leqslant^P_{\text{tt}}$ given by Theorem 1 can now be combined with the definition of $\leqslant^P_{\text{pos}}$ to give a definition of $\leqslant^P_{\text{ptt}}$ that is cleaner than, but equivalent to, the original one that appears in Ladner, Lynch, and Selman (1975).

DEFINITION 3. A set A is *positive truth-table reducible to a set B in polynomial time* ($A \leqslant^P_{\text{ptt}} B$) if there is a positive oracle Turing machine M such that $A \leqslant^P_{\text{pos}} B$ via M and a polynomial time computable function

$f: \{0, 1\}^* \to (c\{0, 1\}^*)^*$ such that, for each input x to M, M only makes queries to B from the list $f(x)$.

Obviously, $A \leqslant^P_{\text{tt}} B$ implies $A \leqslant_{\text{tt}} B$ and $A \leqslant^P_{\text{ptt}} B$ implies $A \leqslant_{\text{ptt}} B$. The following two theorems yield stronger relationships than these, and the stronger relationships will be important for the development to follow. Also, these theorems give an interesting connection between positive machines and positive formulas.

An oracle Turing machine M that operates in polynomial time together with an input word x determines a binary *computation tree* $T = T(M, x)$. The root of T is labeled x. Every leaf is labeled either "accept" or "reject." Every internal node is labeled with a query. Every left branch is labeled "yes" and every right branch is labeled "no", corresponding to whether the state following the query state is YES or NO. A path from the root to a leaf will be called an *accept path* (*reject path*) if the leaf is labeled "accept" ("reject"). $T(M, x)$ is $p(|x|)$ depth bounded, where p is the running time of M.

THEOREM 4 (Ladner, Lynch and Selman, 1975). *If $A \leqslant^P_T B$, then A is truth-table reducible to B in time $2^{p(n)}$ for some polynomial p.*

Proof. Assume $A \leqslant^P_T B$ via M. Given an input word x, a $2^{p(n)}$ time-bounded procedure, where $p(|x|)$ is the depth bound of $T(M, x)$, can write a propositional formula $\varphi(M, x)$: For each accept path τ in $T(M, x)$ write a conjunction of literals C by the rule "if τ branches to the left at an internal node labeled q, then C_τ contains the atom q; if τ branches to the right at an internal node labeled q, then C_τ contains the literal $\sim q$". $\varphi(M, x)$ is the disjunction of each C_τ. If we write $\varphi(M, x)$ as $\varphi(q_1, \ldots, q_k)$, where $q_1, \ldots, q_k$ is a complete list of all the queries that occur in φ, then it should be clear that

$$x \in A \text{ if and only if } \varphi(C_B(q_1), \ldots, C_B(q_k)) = 1.$$

Since $\varphi(M, x)$ can be constructed and evaluated in time $2^{p(|x|)}$, the conclusion follows. ∎

COROLLARY 1. *$A \leqslant^P_T B$ implies $A \leqslant_{\text{tt}} B$.*

THEOREM 5. *An oracle Turing machine M that operates in polynomial time is positive if and only if $\varphi(M, x)$ is a positive formula for every x.*

Proof. In this case C_τ can be defined to be the conjunction of each atom q such that τ branches to the left at an internal node labeled q. The proof in the other direction is obvious. ∎

COROLLARY 2. *$A \leqslant^P_{\text{pos}} B$ implies $A \leqslant_{\text{ptt}} B$.*

3. Semirecursive and P-Selective Sets

Semirecursive sets were invented by Jockusch and studied in his landmark paper (Jockusch, 1968). We call the polynomial time analogue p-selective sets.

Definition 4. A set A is *p-selective* if there is a function $f: \Sigma^* \times \Sigma^* \to \Sigma^*$ so that

(i) f is computable in polynomial time,

(ii) $f(x, y) = x$ or $f(x, y) = y$, and

(iii) $x \in A$ or $y \in A \to f(x, y) \in A$.

The function f is a *selector* for A. If (i) is replaced by (i′) "f is recursive," then by definition A is *semirecursive* (Jockusch, 1968).

The following basic properties hold for both p-selective and semirecursive sets.

Theorem 6. (i) *If A is p-selective, then $\bar{A}$ is p-selective. (If A is semirecursive, then $\bar{A}$ is semirecursive.)*

(ii) *$A \in P$ if and only if $A \leqslant^P_{\text{pos}} \bar{A}$ and A is p-selective. (A is recursive if and only if $A \leqslant_{\text{ptt}} \bar{A}$ and A is semirecursive.)*

(iii) *If $A \leqslant^P_{\text{ptt}} B$ and B is p-selective, then $A \leqslant^P_m B$ and A is p-selective. (If $A \leqslant_{\text{ptt}} B$ and B is semirecursive, then $A \leqslant_m B$ and A is semirecursive.)*

(iv) *If $P \neq NP$, then no $\leqslant^P_{\text{ptt}}$-complete set in NP is p-selective. (No $\leqslant_{\text{ptt}}$-complete r.e. set is semirecursive.)*

The proof of (i) is easy. If f is a selector for A, a selector for $\bar{A}$ is given by $g(x, y) = \text{if } f(x, y) = y \text{ then } x \text{ else } y$.

Properties (ii) and (iii) require arguments similar to but harder than those for the original results. Essentially, the given reduction is used to perform a simulation and p-selectivity is used in clever ways to replace the oracle. Details are given in Selman (1982).

The proof of property (iv) proceeds as follows: First, we prove that if SAT (an encoding of the satisfiability problem) is p-selective, then SAT can be recognized in polynomial time, hence $P = NP$. Let $\varphi(x_1, ..., x_n)$ be a formula of propositional logic with variables $x_1, ..., x_n$. Observe that $\varphi(x_1, ..., x_n)$ is satisfiable if and only if $\varphi(0, x_2, ..., x_n)$ is satisfiable or $\varphi(1, x_2, ... x_n)$ is satisfiable. Assume SAT is p-selective, with selector f. Then, $\varphi(x_1, ..., x_n) \in \text{SAT} \leftrightarrow f(\varphi(0, x_2, ... x_n), \varphi(1, x_2, ..., x_n)) \in \text{SAT}$. Iterate this process, using the result on the right-hand side, until all variables are assigned. Then, $\varphi(x_1, ..., x_n) \in \text{SAT}$ if and only if the final result equals 1. That is our polynomial procedure.

To complete the proof of (iv), apply (iii), where $A = \text{SAT}$ and B is $\leqslant^P_{\text{ptt}}$-complete. The conclusion is that $\text{SAT} \leqslant^P_{\text{ptt}} B$ and B is not p-selective.

The proof of the original result that no $\leqslant_{\text{ptt}}$-complete r.e. set is semirecursive uses the fact that there exist r.e. sets A that are not semirecursive (any simple set that is not hypersimple), and then applies (iii), as above, where B is $\leqslant_{\text{ptt}}$-complete. Techniques of this kind for polynomial time complexity are not known.

The proof of (iv) given above can be extended to yield the following Theorem 7. The following definitions are due to Meyer and Paterson (1979).

DEFINITION 5. A partial order $<$ on Σ^* is OK if and only if

(i) every strictly decreasing chain is finite and there is a polynomial p such that every finite $<$–decreasing chain is shorter than p of the length of its maximum element, and

(ii) $x < y$ implies $|x| \leqslant q(|y|)$, for some polynomial q, and all x and y in Σ^*.

Definition 6. A set A is *self-reducible* if and only if there is an OK partial order and an oracle Turing machine M such that M accepts A in polynomial time with oracle A and, moreover, on any input x in Σ^*, M asks its oracle only about words strictly less than x in the partial order.

We are concerned with sets that are $\leqslant^P_{\text{ptt}}$-self-reducible. (The query machine M in the definition also provides a $\leqslant^P_{\text{ptt}}$-reduction.) As an example, it is easy to see that SAT is $\leqslant^P_{\text{ptt}}$-self-reducible via the reduction used above; $\varphi(x) \in \text{SAT} \leftrightarrow \varphi(0) \in \text{SAT} \vee \varphi(1) \in \text{SAT}$. It is known (Meyer and Paterson, 1979) that the following problems are $\leqslant^P_{\text{ptt}}$-self-reducible:

(i) $\leqslant^P_m$-complete sets in *NP*,

(ii) graph isomorphism, and

(iii) integer factoring (suitably encoded as a recognition problem).

The following theorem is therefore a generalization of Theorem 6 (iv).

THEOREM 7 (Selman, 1982). *If A is $\leqslant^P_{\text{ptt}}$-self-reducible and p-selective, then $A \in P$.*

As a consequence, all of the well-studied combinatorial problems whose feasibility is unknown are not p-selective if $P \neq NP$.

In the theory of recursively enumerable sets all "natural" r.e. sets are either complete or recursive. Thus, the theory of r.e. sets is largely the study of pathologies that elucidate the structure of the r.e. sets. It is not yet known whether all natural sets in *NP* are either *NP*-complete or belong to *P*. But, we can again expect that it is the artificially constructed pathological sets in *NP*

that can illuminate the structure of the class *NP*. Theorem 7 shows that *p*-selective sets in *NP* are pathological.

With this belief, and the properties listed in Theorem 6 in hand, the way to proceed is clear. Construct *p*-selective sets in $NP - P$. This is the task of the next section.

4. The McLaughlin and Martin Construction

It will be useful in this section to give analytic interpretation to strings over $\{0, 1\}$. A dyadic number is a number of the form $m/2^n$, $m \leqslant 2^n$. A word $a_1 \cdots a_n$ over $\{0, 1\}$ denotes the number $a_1/2^1 + \cdots + a_n/2^n$. Thus, Σ^+ represents the set

$$\{m/2^n \mid m, n \in N;\ m \leqslant 2^n\},$$

and, for each n, Σ^n represents the set

$$\{m/2^n \mid m \in N;\ m \leqslant 2^n\}.$$

Observe that $x \leqslant y$ for two dyadic numbers x and y in Σ^+ iff and only if x is less than or equal to y under ordinary dictionary ordering of strings with 0 less than 1. Let r be any real number in the range $0 \leqslant r \leqslant 1$. The *standard left cut* of r is the set $L(r) = \{d \in \Sigma^+ \mid d < r\}$. (Ko (in press) considers left cuts other than standard left cuts.)

If B is a standard left cut of a real number r, then $(a \in B \,\&\, b < a) \rightarrow b \in B$.

LEMMA 1. *If B is an initial segment of some polynomial time computable linear ordering $<_0$ of Σ^*, then B is p-selective.*

Proof. Define $f(x, y) =$ *if* $x \leqslant_0 y$ *then* x *else* y. Since $y \in B$ and $x \leqslant_0 y$ implies $x \in B$, it is easy to see that f is a selector for B. ∎

Since every standard left cut satisfies the hypothesis of Lemma 1, we have as a consequence the following theorem.

THEOREM 8. *Every standard left cut is a p-selective set.*

Relationships between *p*-selective sets and subrecursive complexity-bounded analysis are pursued further in Selman (1981). Also, see Ko (in press, 1982).

We are ready now to see the McLaughlin and Martin construction, as it

appears in Jockusch (1968). Let A be an arbitrary set of positive integers. Define the real number r_A by

$$r_A = \sum \{2^{-i} | i \in A\}.$$

Then, $L(r_A)$ is the standard left cut of r_A.

THEOREM 9 (Jockusch, 1968). *For each recursively enumerable set of integers A,*

(i) $L(r_A) \leqslant_{ptt} A$, *and*

(ii) $A \leqslant_{tt} L(r_A)$.

The following main results of Jockusch follow from the theorem.

COROLLARY 3. (i) *Every r.e. $\leqslant_{tt}$-degree contains an r.e. standard left cut.*

(ii) *No standard left cut is $\leqslant_{\text{ptt}}$-complete.*

(iii) *There exists a standard left cut which is $\leqslant_{\text{tt}}$-complete but not $\leqslant_{\text{ptt}}$-complete.*

Proof. Assertion (i) follows from the theorem, since $\leqslant_{\text{ptt}}$ preserves recursive enumerability. Assertion (ii) follows from Theorem 6 (iv) and Theorem 8. Namely, by Theorem 8, every standard left cut is p-selective, and every p-selective set is trivially semirecursive. But, by Theorem 6(iv), no semirecursive set is $\leqslant_{\text{ptt}}$-complete. Assertion (iii) follows directly from (i) and (ii). ∎

With the reader's forbearance, we will postpone a proof of Theorem 9, for it will follow immediately from Theorem 10 to follow. We desire the analogue of Theorem 9 for polynomial time complexity. The principal difficulty is that $A \leqslant^P_{\text{tt}} L(r_A)$ fails, if integers are given by their usual binary representation. (Proof of this failure exists. See the remark following Corollary 7.) For the complexity results that follow, it is necessary to represent positive integers by tally strings. A tally language is a subset of $\{1\}^*$. Therefore, if we let A be an arbitrary set of positive integers, A is a tally language. The real number r_A is therefore given by

$$r_A = \sum \{2^{-i} | 1^i \in A\}.$$

Whereas any two representations of integers are recursively equivalent, succinctness of representation does seem to affect the complexity of a set of integers. Indeed, a tally language A belongs to NP if and only if the

corresponding language of binary strings belongs to NEXT. Containment of tally languages in $NP - P$ is a reasonable hypothesis, and is equivalent to the assertion that DEXT $\neq$ NEXT (Book, 1974). (The open problem DEXT $\neq$ NEXT is closely related to the spectrum problem of first order logic (Jones and Selman, 1974).) Also, $P = NP \rightarrow$ DEXT = NEXT, (Book, 1974; Jones and Selman, 1974).

THEOREM 10 (Selman, 1982). *For every tally language A, there exist sets B and C such that*

(i) $B \leqslant^P_{\text{ptt}} A$,
(ii) $A \leqslant^P_T B$,
(iii) $C \leqslant^P_{\text{tt}} A$,
(iv) $A \leqslant^P_T C$,
(v) $B \leqslant^P_{\text{ptt}} C$,
(vi) $C \leqslant^P_{\text{tt}} B$,
(vii) *B is p-selective,*
(viii) *If C is p-selective, then $C \in P$.*

Proof. Let $B = L(r_A)$. Then, (vii) holds because B is a standard left cut. Define $C \subseteq \{0, 1\}^*$ so that for each $n \geqslant 1$, C contains exactly one string of length n: The string $x = x_1 \cdots x_n$ is to belong to C is and only if:

$$x_i = 1, \qquad \text{if} \quad 1^i \in A,$$

and

$$x_i = 0, \qquad \text{otherwise } (i = 1,..., n).$$

C is of course the set of all initial segments of the binary expansion of r_A. It should be clear that $B = \{x \mid \exists y[\, y \in C \,\&\, x \leqslant y]\}$.

The following algorithm demonstrates (i) $B \leqslant^P_{\text{ptt}} A$. Let $x = x_1 \cdots x_n$ in $\{0, 1\}^*$ be an input string.

```
i:= 1;
repeat
  if x_i = 1 and 1^i ∉ A
    then reject input and halt;
  if x_i = 0 and 1^i ∈ A
    then accept input and halt;
  i:= i + 1
until (accept or reject or i = n);
if x_n = 0 or 1^n ∈ A
  then accept else reject;
```

The reduction is easily seen to be positive. Rejection is only possible if a symbol $x_i = 1$ and the corresponding string 1^i does not belong to the oracle set. Therefore, the addition of strings 1^i to the oracle set cannot cause previously accepted strings to become rejected. Further, the set of all possible queries $\{1, 1^2, ..., 1^n\}$ can be listed in advance. Thus, by Theorem 1, we have $B \leqslant^P_{\text{ptt}} A$.

$A \leqslant^P_T B$ is given by the algorithm: Let 1^n be input. Generate the largest string x of length n of B. Accept 1^n if and only if the rightmost symbol of x is 1.

Part (viii) is proved as follows: Let f be a selector for C. Compute $f(0, 1)$. Since exactly one of 0 or 1 belongs to C, $f(0, 1) \in C$. Let $a_1 = f(0, 1)$. Compute $f(a_1 0, a_1 1)$. Since exactly one of $a_1 0$ or $a_1 1$ belongs to C, $f(a_1 0, a_1 1) \in C$. Let $a_1 a_2 = f(a_1 0, a_1 1)$. Continue in this manner. A string x of length n belongs to C if and only if $x = a_1 ... a_n$. Thus, $C \in P$.

The remaining parts (iii)–(vi) are left as easy exercises for the reader, with solutions to be found in Selman (1982), Theorem 12; (1979, Theorem 5). ∎

Let us observe that the proof of Theorem 10 breaks down in case $A \subseteq \{0, 1\}^*$. Consider the algorithm used to prove part (ii), $A \leqslant^P_T B$. On input 1^n this algorithm needs to query strings of length n. If n were presented as a binary word $b(n) \in \{0, 1\}^*$ then $|b(n)| \approx \log_2 n$. Hence, no polynomial time algorithm on input $b(n)$ could query strings of length n.

To see that Theorem 9 is a special case of Theorem 10, we make the following observations. Let A be an arbitrary r.e. set (presented in unary notation). By Theorem 10 (i), $L(r_A) \leqslant_{\text{ptt}} A$; by Corollary 1, part (ii), $A \leqslant^P_T L(r_A)$, becomes $A \leqslant_{tt} L(r_A)$. To repeat, what is needed to make the McLaughlin and Martin construction work at the polynomial time complexity level is not a modification of the construction per se. All that is required is care about the presentation of the input to the construction.

The following corollaries are the main complexity applications of Theorem 10.

COROLLARY 4. *Let A be any tally language not in P. Then, there exist $\leqslant^P_T$-equivalent sets B and C such that $C \leqslant^P_{\text{tt}} B$, but $C \not\leqslant^P_{\text{ptt}} B$. Also, $\bar{B} \not\leqslant^P_{\text{pos}} B$. Therefore, we prove $\leqslant^P_{\text{tt}} \neq \leqslant^P_{\text{ptt}}$ and $\leqslant^P_{\text{tt}} \neq \leqslant^P_{\text{pos}}$ without any use of diagonalization, except, of course, for the standard diagonalization that is implicit in obtaining A not in P.*

COROLLARY 5. *If* DEXT $\neq$ NEXT, *then there exist sets B and C such that*

(i) $B \in NP - P$,

(i) $B \leqslant^P_{\text{ptt}} C$,

(iii) $C \leqslant_{\mathrm{tt}}^{P} B$,

(iv) $C \not\leqslant_{\mathrm{ptt}}^{P} B$, *and*

(v) $\bar{B} \not\leqslant_{\mathrm{pos}}^{P} B$.

Thus, if DEXT $\neq$ NEXT, *there exists an* $NP \leqslant_{\mathrm{tt}}^{P}$*-degree not equal to* P *which consists of at least two* $\leqslant_{\mathrm{pos}}^{P}$ *– degrees. In addition, there exists an* $NP \leqslant_{\mathrm{ptt}}^{P}$*-degree which consists of a single* $\leqslant_{m}^{P}$*-degree.*

It would be a stronger result to cause $\leqslant_{\mathrm{ptt}}^{P}$ and $\leqslant_{\mathrm{tt}}^{P}$ to differ on *NP*, so that both of the sets B and C belong to *NP*. The following corollary is due to a lovely observation made by C. B. Wilson (1980).

COROLLARY 6. *Assume* NEXT $\cap$ coNEXT $\neq$ DEXT. *Then there exist sets* B *and* C *in* $NP - P$ *such that* $B \leqslant_{\mathrm{ptt}}^{P} C$ *and* $C \leqslant_{\mathrm{tt}}^{P} B$, *but* $C \not\leqslant_{\mathrm{ptt}}^{P} B$.

Proofs follow readily. Once again, they can be found by the interested reader in Selman (1982).

Our next, and final, result is not only interesting in its own right, it also demontrates that the collapse of Theorem 10 when applied to binary languages cannot be remedied. For this exposition, we utilize the following result of Karp and Lipton (1980): No tally language can be *NP*-hard unless the polynomial hierarchy collapses to Σ_2^P.

THEOREM 11 (Selman, 1981). *For each standard left cut* B *there is a tally language* A *such that* $A \leqslant_{\mathrm{tt}}^{P} B$ *and* $B \leqslant_{\mathrm{ptt}}^{P} A$.

Proof. If B is the standard left cut of a real number x, define $A \subseteq \{1\}^*$ so that $1^n \in A$ if and only if the n-th bit of the binary expansion of x is 1. Then $x = r_A$ and $B = L(r_A)$. Hence, Theorem 10 applies. ∎

COROLLARY 7. *No standard left cut can be* $\leqslant_{\mathrm{tt}}^{P}$*-complete in NP unless the polynomial hierarchy collapses to* $\sum_2^P$.

Thus, we have reached the limit of applicability of the left cut construction. Parts (i) and (iii) of Corollary 3 cannot be true for polynomial time-bounded reducibilities (unless, of course, the polynomial hierarchy collapses). Furthermore, were Theorem 10 true for languages $A \subseteq \{0, 1\}^*$, then application of this theorem in the case A is *NP*-hard, would yield an *NP*-hard standard left cut B. However, using Theorem 11 and the Karp–Lipton result, this too is impossible unless the polynomial hierarchy collapses to $\sum_2^P$.

5. DISCUSSION

We have seen that there are intimate relationships between effective reducibilities and polynomial time reducibilities and between semirecursive

sets and p-selective sets. We have seen that polynomial time reducibilities can be distinguished on NP using the notion of p-selective sets. We would certainly prefer to have these results with hypothesis $P \neq NP$ than with the hypothesis DEXT $\neq$ NEXT.

Our technique is lifted from analogous results in recursion theory. We have seen where these analogies between recursion theory and complexity theory work, and where and how they break down. They do not work to obtain results about complete sets in NP. However, just what is the connection between p-selective sets and standard left cuts? Perhaps p-selective sets can still be used to distinguish, say, $\leqslant_m^P$ – completeness from $\leqslant_T^P$-completeness in NP. If this is to be so, then new techniques will be required first, in order to construct p-selective sets in NP that are not also standard left cuts.

RECEIVED: June 15, 1981; REVISED: June 4, 1982

REFERENCES

BAKER, T., J. GILL, AND R. SOLOVAY (1975), Relativizations of the $P = ?$ NP question, *SIAM J. Comput.* **4** 431–442.

BOOK, R. (1974), Tally languages and complexity classes, *Inform. Contr.* **26**, 186–193.

BOOK, R. (1974), Comparing complexity classes, *J. Comput. System Sci.* **9**, 213–229.

BREIDBART, S. (1978), On Splitting recursive sets, *J. Comput. System Sci.* **17**, 56–64.

COOK, S. A. (1971), The complexity of theorem-proving procedures, *in* "Third Annual ACM Symposium on Theory of Computing."

GAREY, M. R. AND JOHNSON, D. S. (1979), Computers and Intractability: A Guide to the Theory of *NP*-Completeness, W. H. Freeman and Co., San Francisco.

JOCKUSCH, C. G., JR. (1968), Semirecursive sets and positive reducibility, *Trans. Amer. Math. Soc.* **131**, 420–436.

JONES, N., AND A. L. SELMAN (1974), Turing machines and the spectra of first-order formulas, *J. Symbol. Logic* **29**, 139–150.

KARP, R. M. (1973), Reducibility among combinatorial problems, *in* "Complexity of Computer Computations" (Miller and Thatcher, eds.). Plenum Press, New York.

KARP, R. M. AND LIPTON, R. (1980), Some connections between nonuniform and uniform complexity classes, *in* "Twelfth Annual ACM Symposium on Theory of Computing."

KO, K., On the definitions of some complexity classes of real numbers, *Math. Systems Theory*, in press.

KO, K. (1982), The maximum value problem and NP real numbers, *J. Comput. System Sci.* **24**, 15–35.

LADNER, R. E. (1975), On the structure of polynomial time reducibility, *J. Assoc. Compat. Mach.* **22**, 155–171.

LADNER, R. E., N. A. LYNCH, AND A. L. SELMAN, (1975), A comparison of polynomial time reducibilities, *Theor. Comput. Sci.* **1**, 103–123.

LONG, T. J., Strong nondeterministic polynomial time reducibilities, *Theor. Comput. Sci.*, in press.

MEYER, A., AND PATERSON, M. (1979), With what frequency are apparently intractable problems difficult?, MIT Technical Report.

POST, E. L. (1944), Recursively enumerable sets of integers and their decision problems, *Bull. Amer. Math. Soc.* **50**, 284–36.

SELMAN, A. L. (1979), *P*-selective sets, tally languages, and the behavior of polynomial time reducibilities on *NP*, *Math. Systems Theory* **13**, 55–65.

SELMAN, A. L. (1982), Reductions on *NP* and *p*-selective sets, *Theor. Comput. Sci.* **19**, 287–304.

SELMAN, A. L. (1981), Some observations on *NP* real numbers and *p*-selective sets, *J. Comput. System Sci.* **23**, 326–332.

STOCKMEYER, L. J. (1976), The polynomial-time hierarchy, *Theor. Comput. Sci.* **3**, 1–22.

WILSON, C. (1980), "Relativization, Reducibilities, and the Exponential Hierarchy," Department of Computer Science Technical Report 140/80, University of Toronto.

WRATHALL, C. (1976), Complete sets and the polynomial hierarchy, *Theor. Comput. Sci.* **3**, 23–33.